Numerical Simulation on the SiCortex Supercomputer Platform: a Preliminary Evaluation

Vincent Heuveline, Björn Rocker, Staffan Ronnas

Universität Karlsruhe (TH) - Karlsruhe Institute of Technology (KIT)
Institute for Applied and Numerical Mathematics IV
RG Numerical Simulation, Optimization and High Performance Computing,
76128 Karlsruhe, Germany

vincent.heuveline@kit.edu
bjoern.rocker@kit.edu
staffan.ronnas@student.kit.edu

Abstract. Most cluster systems used today for high-performance scientific computing are built from off-the-shelf standard components placed in racks. SiCortex has chosen a different strategy and offers a line of integrated cluster machines based on a customized low-frequency MIPS multicore processor and a specialized network fabric.
We investigate the potential of the SiCortex platform for numerical simulation by analyzing the performance of a set of elementary benchmarks and two fluid dynamics applications executed on the SC072 and the SC5832 systems. The elementary benchmarks quantify the performance in terms of computation rate, memory bandwidth and communication latency. The fluid dynamics applications provide insight into how well existing scientific code performs on the system. The results are compared to those obtained on a commodity cluster with Intel Xeon cores and Infiniband interconnect. The focus of the evaluation is computational performance, but we also consider the energy consumption for all three machines.
Our results indicate that while the SiCortex systems might be well suitable for applications that can be parallelized to a very fine level, they are outperformed by commodity clusters when this is not the case. However, an analysis of the CFD applications shows that the SiCortex systems make it possible to significantly reduce the energy consumption compared to a commodity cluster.

1 Introduction

During the past years systems composed of off-the-shelf components have dominated the market for high performance computing clusters. A company building machines in a different way is SiCortex with the products SC072 and SC5832. These systems are based on low-frequency MIPS64 cores and use a customized

interconnect for communication. The idea is to provide a machine that better balances the computation rate of the CPU and the memory bandwidth by spreading the computations over a very large number of cores. For applications that can be parallelized with a fine granularity, the low-power nodes are thought to have the potential to also reduce the energy costs of large scientific computations.

This paper presents the results of several benchmarks performed on the SC072, the SC5832, and a commodity cluster based on Intel Xeon cores and Infiniband interconnect. The benchmarks include both elementary kernels and two full applications that solve a problem in fluid dynamics.

A description of the hardware and software used for the tests is given in Sections 2 and 3. The results of the benchmarks are presented in Sections 4 and 5. Section 6 summarizes our analysis of these results.

2 Hardware Description

This section gives an overview of the three cluster systems used in our tests.

The **SiCortex SC5832** is a single cabinet machine with 36 circuit modules each accommodating 27 SiCortex node chips and associated memory [17]. Each node hosts a six-core MIPS64 processor with a clock rate of 700 MHz. The most important specifications of the machine used for the test in this paper are listed in Table 1.

All nodes on the SC5832 can communicate via a customized interconnect, which is based on a Kautz-graph topology of degree 3 and diameter 6. In this network topology, there are 3 disjoint paths from each node to any other node, which provides fault tolerance in the case that a node or link fails. A maximum of 6 hops are required to transmit a message between any two nodes.

The **SC072** is the desktop version of the bigger SiCortex machine and contains 12 nodes with 72 cores in total. The diameter of the Kautz-graph with degree 3 is 2 for this machine. The machine used for the test is equipped with 8 GB of main memory per node. A performance analysis of an earlier version of the SC072 is available in [11].

The commodity cluster used for our comparison is the **Institutscluster IC1** located at the Steinbuch Centre for Computing (SCC) [5] at the University of Karlsruhe / Karlsruhe Institute for Technology (KIT) [14]. It consists of five cabinets containing a total of 200 computing nodes, each equipped with two Intel quadcore EM64T Xeon 5355 processors with Cloverton architecture running at 2.667 GHz. The interconnect used is Infiniband 4x DDR.

Table 1 summarizes the characteristics of these three systems. Most values come from the specifications given by the respective manufacturers. The values for the power consumption come from three sources: for the SC072, the power was measured with a wattmeter by the authors; for the SC5832, the power was taken from a webpage of the University of Magdeburg [16]; and for the IC1, the values were communicated from the SCC. The values of the Linpack benchmark computation rate come from our own measurement (SC072), the HPCC results database (SC5832) and the website of the SCC (IC1).

	SC072	SC5832[a]	IC1[b]
Nodes	12	972	200
Processors per node	1	1	2
Cores per processor	6	6	4
Theoretical comp. rate / core	1.4 GFlop/s	1.4 GFlop/s	10.7 GFlop/s
Theoretical comp. rate / node	8.4 GFlop/s	8.4 GFlop/s	85.3 GFlop/s
Theoretical comp. rate full machine	100.8 GFlop/s	8.2 TFlop/s	17.6 TFlop/s
Linpack performance full machine	56.6 GFlop/s	4.73 TFlop/s[c]	15.2 TFlop/s
L2-cache per processor	1.5 MB	1.5 MB	8 MB
Memory per node	8 GB	4 GB	16 GB
Memory full machine	92 GB	4 TB	32 TB
Memory bandwidth per node	10.7 GB/s	10.7 GB/s	10.7 GB/s
Power consumption load	330 W	21 kW	103 kW
Power consumption idle	230 W	11 kW[d]	56 kW

[a] The evaluated machine is located at the computing center of the University of Magdeburg. (See [16])
[b] See http://www.rz.uni-karlsruhe.de/ssck/ic.php
[c] See [8]
[d] See [16]

Table 1. Key system characteristics of the three clusters used for the tests.

3 Software Description

The benchmarks that were used fall into two categories: elementary kernel benchmarks, which aim to measure an isolated aspect of the performance of the machine, such as its floating point computation rate, its memory bandwidth or its interconnect communication speed; and a computational fluid dynamics (CFD) application benchmark, which gives an indication of how well the systems perform on typical scientific simulations. This section presents an overview of the software used to test the machines.

On the IC1 the Intel compiler version 10.1.022 was used together with the Intel MKL 10.1 numerical library. On the SiCortex machines we used the Pathscale compiler and ATLAS BLAS 3.7.32 for the benchmarks.

3.1 Elementary Benchmarks

HPCC Suite The HPC Challenge Benchmark is a suite of seven benchmark kernels created by the DARPA HPCS program[4]. Together they provide a more balanced view of the performance of general-purpose HPC systems than the classical Linpack benchmark, whose performance is limited mainly by the rate of floating point arithmetic that the processors are capable of. In the present work, we concentrate on two of the benchmarks: the MPI ping-pong latency and the STREAM **triad** tests. Full benchmark results for the SC5832 are available in the HPCC results database [8].

LLCbench The LLCbench benchmark collection consists of three parts, aiming to measure the performance of simple linear algebra functions (Blasbench, [12]), the performance of the memory hierarchy (Cachebench, [13]) and the speed of the communications interconnect (MPbench). This report includes results obtained with the first two of these benchmarks.

3.2 Application Benchmarks

The numerical simulation test case is a standard example in fluid dynamics: 3D stationary lid-driven cavity on a cuboid. This problem is solved with the finite element software HiFlow and the Lattice-Boltzmann software OpenLB.

HiFlow The HiFlow package is a parallel, finite element library with a strong emphasis on computational fluid dynamics and is written in C++. The library includes the following features:

- CFD (incompressible Navier-Stokes, Low-Mach flows, heat convection)
- Reactive flows
- Eigenvalue computation for stability analysis
- Conforming h- and hp-FEM
- A posteriori error estimation for FEM
- Moving boundaries

HiFlow uses the PETSc [1–3] and METIS [9] libraries for solving this linear system in parallel on all three machines.

OpenLB The OpenLB project [15] provides a publicly available C++ library mainly intended for researchers and engineers who simulate fluid flows by means of lattice Boltzmann methods [10]. The library includes the following features:

- Flows in complex geometries
- Turbulent flows
- Multiphase flows
- Thermal flows

The library enables a fast implementation of both simple applications and advanced CFD problems. It is easily extensible to take into account new physical models.

The used version of OpenLB is capable of hybrid parallelisation using MPI for internodal communication and OpenMP within the nodes. For a deeper analysis of the parallelisation see [7].

4 Elementary Benchmark results

4.1 HPCC MPI-latency Results

The MPI latency benchmark from the HPCC benchmark suite measures the latency of the communication between two cores by sending messages from one core to another. The message size for the test is 8 bytes. All pairs of cores are used in the test, and the minimum, maximum and average values of the latency are reported. The test was run using different number of nodes, with either one or all cores per nodes active, and the results are shown in Fig. 1.

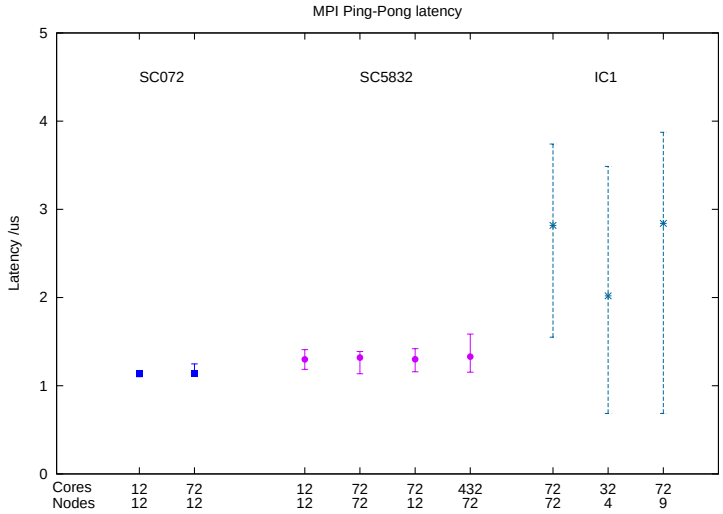


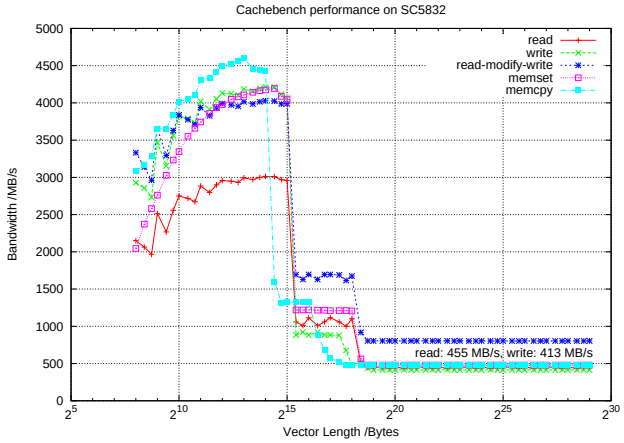
Fig. 1. MPI Ping-Pong latency. The graph shows the average and the variation around this value.

The spread of the latency on the SC072 machine is very small due to the fact that either one or two hops are required for communication between any two cores. On the SC5832 where the topology has diameter 6, the spread is a little larger. Much larger fluctuations are seen on the IC1, which has a fat-tree topology. [6] provides more information on network topologies and their impact on communication performance.

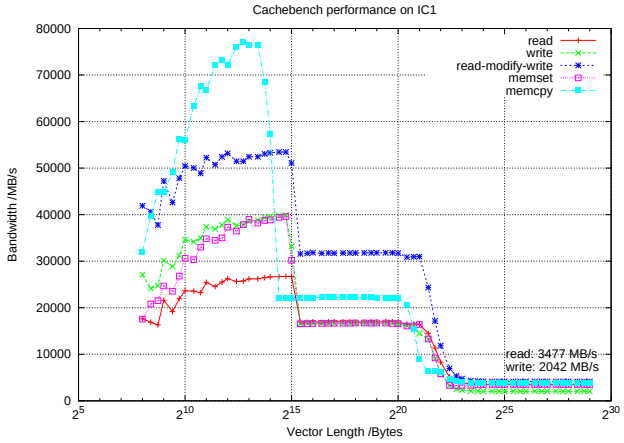
When all cores per node are used, the IC1 has a lower minimum latency than the SiCortex machines. A comparison with the test with one core per node shows that this minimum corresponds to communication within a single node. The average latency on the IC1 is circa 2 μs with 4 nodes and almost 3 μs with 9 nodes. On the SiCortex machines, the average latency with both 12 and 72 nodes is less than 1.5 μs, which indicates a better scalability with respect to this metric than the IC1.

4.2 Cachebench Results

The Cachebench benchmark was run on a single core on each of the three clusters. The results for the SC5832 and the IC1 are shown in Fig. 2(a) and 2(b). The corresponding graph for the SC072 has been omitted since the results were practically identical to those of the SC5832.



(a) SC5832 results



(b) IC1 results

Fig. 2. Performance of memory hierarchy on the SC5832 and IC1 machines. The results are similar for the SC072 and the SC5832.

The effect of the use of cache memory on both machines is obvious from the graphs. The L1-cache of both processors has the same size (32 kB) but the bandwidth on the Xeon processor is approximately 8 to 10 times higher on the

read and write operations. The L2-cache, which in contrast to the L1-cache is not exclusive for each core, is considerably larger on the Xeon processors. It can be shared dynamically on the Intel processor, but not on the SiCortex, which leads to a steeper decrease in bandwidth on the latter system when the data cannot be held in this cache memory.

For arrays which fit into the L1 cache on the IC1, the **memset** and **memcpy** operations perform considerably better than the hand-coded write and read-modify operations; while on the SC5832, the difference in performance is minor. Clearly, the system calls have been tuned to take advantage of the special features of the Intel architecture, while the same is not true for the SiCortex MIPS platform. For large arrays, the hand-coded versions are equivalent or better than the library calls.

For applications that intensively access the memory, the most important information that can be extracted from the graphs is the bandwidth to the main memory, which can be read at the far right of each graph. The bandwidth on the IC1 is approximately 7.5 times higher than that of the SC5832 for the read operation, and 4.9 times higher for the write operation.

4.3 HPCC STREAM Results

The STREAM benchmark evaluates the memory performance within one node by performing a vector operation on an array of data that does not fit into the L2 cache. The test has two different modes: "single" mode, where the test is run only on one randomly chosen core; and "star" mode, where all active cores run the test. The star mode makes it possible to measure the degradation of the memory bandwidth when all cores on a node require access to the memory controller simultaneously.

The graph in Fig. 3 shows the result of the **triad** ($z = \alpha x + y$) operation for different numbers of nodes. In each configuration either one or all cores were active on each node. When only one core is active on each node, the single and star modes give the same results, as one would expect.

On the IC1 cluster, the measured bandwidth is reduced drastically when going from the single mode to the star mode, whereas on the SC072 and the SC5832 this reduction is much less pronounced. This suggests that the memory bandwidth is less likely to be a bottleneck when scaling an application from one to six cores per node on the SiCortex systems than on the IC1. On the latter system, the time of execution of a parallel application can often be reduced by spreading the processes over more nodes and using fewer cores on each node, in order to increase the available memory bandwidth. The results of the STREAM benchmark suggest that the same is not likely to be true on the SiCortex machine.

However, it should not be forgotten that even when all cores are used, the measured bandwidth is 50% higher on the IC1 than on the SiCortex machines.

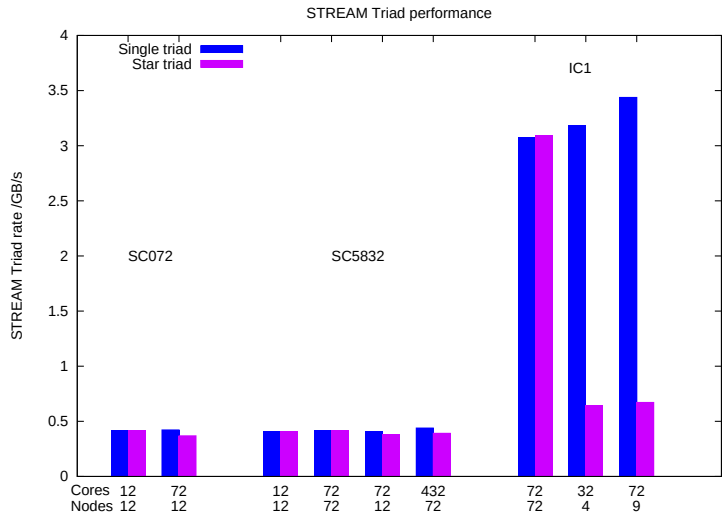


Fig. 3. Single- and Star-STREAM triad performance for SC072, SC5832 and IC1.

4.4 Blasbench Results

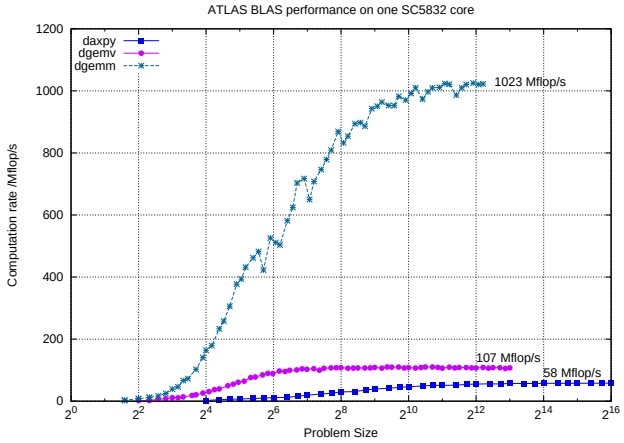
The Basic Linear Algebra Subroutines (BLAS) are a central part of many scientific computing codes. The Blasbench testsuite measures the single-core computation rate of a kernel from each of the three levels of the BLAS. These kernels are **daxpy**, **dgemv**, and **dgemm**, which respectively compute vector-vector, matrix-vector and matrix-matrix multiplications.

Fig. 4(a) and 4(b) show the results of the Blasbench test on the SC5832 and IC1, respectively. The graph for the SC072 is omitted since it is identical to that of the SC5832.

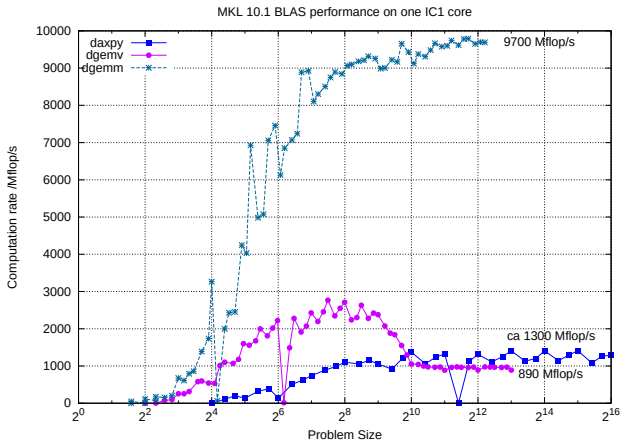
All three kernels show an increase in computation rate with the characteristic size of the problem N up to a certain limit, where the curves flatten out. On the IC1, the performance is higher for problem sizes up to $N = 1024$, since the data can fit into the L2 cache. On the SC5832, this effect is absent, which suggests that the ATLAS BLAS implementation does not make efficient use of the cache.

Overall, the **daxpy** operation is over 22 times faster on the IC1 than on the SiCortex machines, and the **dgemv** operation is 8 times faster. For **dgemm**, the IC1 is 9.5 times faster, which should be compared to the ratio 7.6 in theoretical peak computation rate between the two platforms. The IC1 achieves circa 90% of its peak performance with the **dgemm** kernel, while the SiCortex systems reaches only about 70% efficiency. It seems clear that the highly optimized Intel MKL implementation on the Intel processors significantly outperforms the ATLAS BLAS library on the SiCortex MIPS64 processors.

It should however be noted that the IC1 shows some irregular but reproducible variations with the problem size which are not present on the SiCortex systems.



(a) SC5832 results



(b) IC1 results

Fig. 4. Single core BLAS performance on the SC5832 and the IC1. The performance on the SC072 and the SC5832 is identical.

5 Application Benchmark Results

5.1 Test case for HiFlow

The test case for the CFD package HiFlow is a standard example: 3D lid-driven cavity (LDC) on a cuboid. The geometry is uniformly refined to 65536 cells and the incompressible Navier-Stokes equations are solved for a stationary solution with Q2 elements for the velocity and Q1 elements for the pressure. The number of unknowns in the resulting linear system of equations is approximately 1.7 million.

Fig. 5 shows the time taken to assemble and solve the lid-driven cavity problem with different number of processes distributed in different ways over the nodes on the three clusters. For all three machines, there is a point after which the gain in performance for each added process starts decreasing. This phenomenon, which is typical when a fixed problem size is used, is due to Amdahl's law, which states that the gain in speed from parallelization is limited by the fraction of time spent in sequential code.

Overall, the slope of the curves is similar for the SiCortex machines and the IC1 up to approximately 128 processes, at which point the performance increase with each added process falls off rapidly on the SiCortex systems.

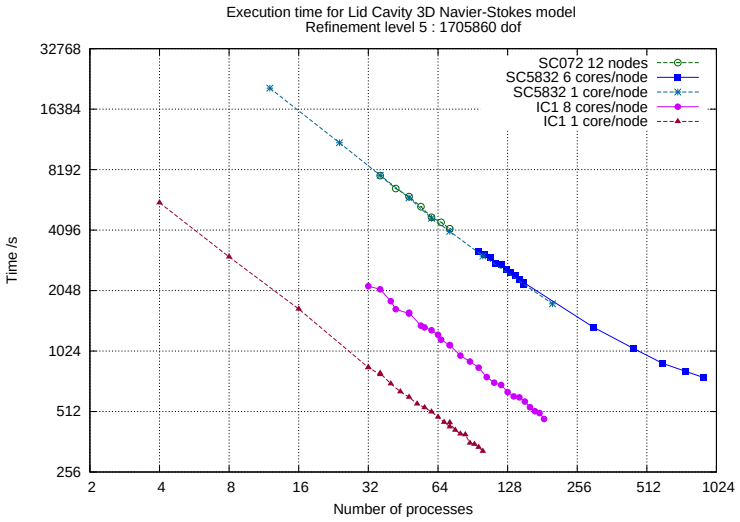


Fig. 5. Execution time for 3D lid-driven Cavity with HiFlow on the IC1, SC072 and SC5832. Incompressible Navier Stokes equations are solved with 1,705,860 degrees of freedom.

As was seen for the STREAM benchmark in Section 4.3, this test shows the impact of the memory bandwidth limitation on the IC1 through the fact that the execution with 1 core per node is more than twice as fast as with 8 cores per node. No such difference exists on the SiCortex systems, which can be interpreted as if the SiCortex systems have a better balance between computation rate and memory bandwidth. The fact remains, however, that the execution time on the IC1 is much lower than on the SiCortex machines. As an example, with 96 processes, the IC1 solves the problem 9.4 times faster than the SC5832 with one core per node and 3.8 times faster with eight cores per node. These performance ratios are similar also with other numbers of processes.

The main promise of the SiCortex systems is not computation speed but rather efficiency in terms of energy. It is therefore interesting to investigate this

aspect of the cluster systems. By deducing the power consumption P under load per node from the values given in Table 1, we obtain 514 W per node on the IC1, 21 W on the SC5832 and 28 W on the SC072. With these values, it is possible to estimate the total amount of energy E that is been spent for a computation which takes t seconds through the relation $E = P \cdot t$.

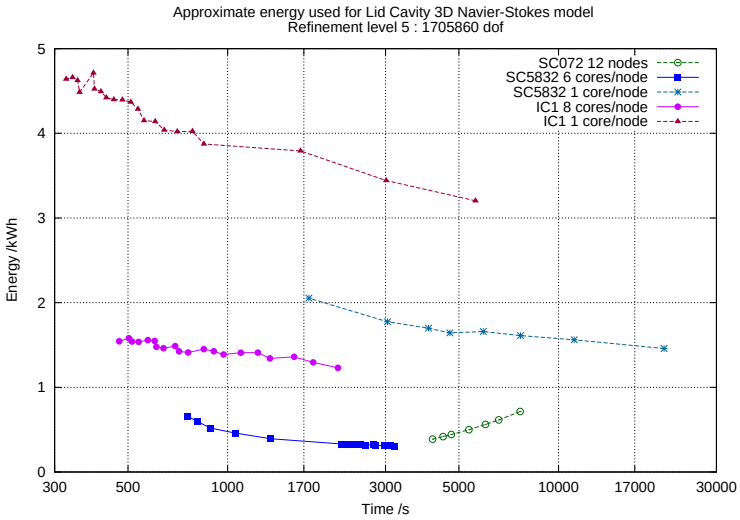


Fig. 6. Energy consumption in relation to the computation time for solving the Navier-Stokes equations on a cube with finite elements. The problem size is constant and leads to a linear system with 1.7 million unknowns. On the IC1 and SC5832, configurations with one and all cores per node were tested; while on the SC072 twelve nodes were always used, with different number of cores per node.

Fig. 6 shows the energy consumption as a function of the time taken to solve the problem. As long as the execution time scales perfectly with the number of cores, the curves are flat, since the increase in power from using more cores is compensated by a corresponding decrease in execution time. Using one core per node on the IC1 clearly makes it possible to solve the problem in the shortest time, but the energy cost is very high. To lower the energy consumption, all cores should be used on each node. For execution times over 1300 s, where the scaling is good on the SC5832, the energy consumption for a fixed execution time on this machine is between 3 and 4 times lower than on the IC1. Hence, if one can afford to let the computation take a longer time to finish, large energy savings are possible on the SiCortex system.

5.2 Test case for OpenLB

The test case for the Lattice Boltzmann/CFD package OpenLB was again the 3D lid-driven cavity simulation, this time with an instationary flow on a cubic geometry. A Lattice Boltzmann method with D3Q19 discretization model was used as explained in [7]. In order to decrease the execution time of the tests, only the first twenty time steps were computed, although a typical simulation would use many more time steps. The execution time was therefore dominated by the time to initialize the data structures; whereas in a real computation this time would be negligable. Hence the initialization time was not taken into account for the time measurements.

The size of the problem was scaled with the number of processes used so that when the scaling is perfect, the computation time should stay constant. The problem size was chosen differently on IC1 and SC5832, in order to obtain approximately the same execution time on both systems.

Disregarding the initialization time, both the execution time and memory consumption of the program scale as $\mathcal{O}(\frac{N^3}{p})$ where N is the number of discretization points in each dimension and p the number of processes. In order to keep the execution time T constant when varying p, the following relation was used to determine the size of the problem: $N = \lfloor \alpha \sqrt[3]{p} \rfloor$. For the results shown in Fig. 7, $\alpha = 100$ was used on the SC5832 and $\alpha = 180$ was used on the IC1, which makes the latter problem almost 6 times bigger than the former.

The results achieved with OpenLB are representative for explicit methods which are usually limited by the available memory bandwidth.

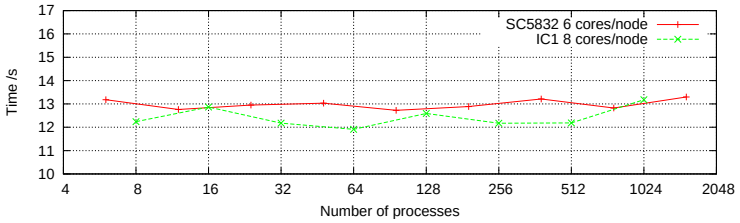


Fig. 7. Execution time for 20 timesteps of 3D lid-driven Cavity with OpenLB on the IC1 and SC5832 without initialization time. The problem size was scaled with the resources. The size of the problem is scaled according to $N = \lfloor \alpha \sqrt[3]{p} \rfloor$. $\alpha = 100$ on the SC5832, and $\alpha = 180$ on the IC1.

The results in Fig. 7 show that the computation time stays approximately constant, which indicates that this code scales well on both machines.

6 Conclusion

The results of the benchmarks have given some insight into the characteristics of the integrated and custom-designed SiCortex SC072 and SC5832 cluster systems. Their performance has been compared to that of the IC1, which is a system assembled from off-the-shelf components from different vendors.

In terms of floating-point computation rate for each core, the SiCortex systems are clearly inferior to the IC1. Sequential parts of an application thus have a larger risk to become a limiting factor on the SiCortex system than on the commodity cluster.

The ability to fully exploit multicore processors is often limited by the bottleneck associated with access to the main memory. The HPCC STREAM results show that this bottleneck has been removed on the SiCortex systems, whereas it is very significant on the IC1. This observation is confirmed through the CFD application benchmark, where the IC1 exhibits a large difference in execution time when only one core per node is used instead of all cores. On the SiCortex machines, all cores per node can be used without performance degradation.

On the other hand, the absolute performance of the SiCortex nodes is somewhat disappointing: the IC1 performs 50% better in the HPCC STREAM bench-

mark, and is 8 times faster on the BLAS **dgemv** operations. There seems to be room for improvements both of the hardware and the software.

When a full node is used, the IC1 also outperforms the SiCortex systems, but the ratio in performance is usually smaller than for a single core, even though the number of cores per node is higher on the IC1. One reason for this is that the memory bandwidth becomes a limitating factor on the IC1 to a larger extent than on the SiCortex machines. This is illustrated by the results of the HPCC STREAM benchmark, which indicate that the SiCortex cores are so slow that common scientific computations are not limited by the bandwidth of the memory.

In order to achieve comparable execution times with a given application on the slower processors in the SiCortex system, it is necessary to spread the computation over a larger number of nodes. This is made possible by the low latency of the system's network. The potential for exploiting a large number of cores is also determined by the scalability of the software. For the tested parallel fluid dynamics codes the SC5832 performs well compared to the IC1. Our finite element CFD application requires 4 to 5 times more cores on the SiCortex machine than on the IC1 to achieve the same computation time, when all cores per node are used. The Lattice-Boltzmann approach also scales very well.

The SiCortex machines can also be used in a more energy efficient way than the x86 cluster. For a fixed computation time, an execution of the finite element application on the commodity cluster consumes between 3 and 4 times more energy, when all cores per node are used. In view of current environmental concerns, this makes the SiCortex platform a very interesting alternative for tasks that are not time-critical.

7 Acknowledgements

This evaluation was only possible thanks to the cooperation with the University Computing Center of the Otto von Guericke University in Magdeburg. We especially want to thank Dr. Joerg Schulenburg for his assistance.

References

1. Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Users Manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.
2. Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2009. http://www.mcs.anl.gov/petsc.
3. Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

4. J. Dongarra and P Luszczek. Introduction to the HPC Challenge benchmark suite. Technical Report UT-CS-05-544, University of Tennessee, 2005.
5. Steinbuch Center for Computing. `http://www.scc.kit.edu`.
6. David Groth and Toby Skandier. *Network+ Study Guide*. Sybex Inc., 2005.
7. V. Heuveline, M.J. Krause, and J. Latt. Towards a hybrid parallelization of lattice boltzmann methods. *Computers and Mathematics with Applications*, 58:1071–1080, 2009.
8. HPCC. HPCC Results Database. `http://icl.cs.utk.edu/hpcc/hpcc_results.cgi`.
9. George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.
10. LBMethod.org. `http://www.lbmethod.org`.
11. Brian J. Martin, Andrew J. Leiker, James H. Laros III, and Doug W. Doerfler. Performance Analysis of the SiCortex SC072. The 10th LCI International Conference on High-Performance Clustered Computing, 2009.
12. Philip J. Mucci, Kevin London, and John Thurman. The CacheBench Report. Technical report, Innovative Computing Laboratory, University of Tennessee, 1998.
13. Philip J. Mucci, Kevin London, and John Thurman. The BLASBench Report. Technical report, Innovative Computing Laboratory, University of Tennessee, 1999.
14. Karlsruhe Institute of Technology. `http://www.kit.edu`.
15. OpenLB. `http://www.openlb.org`.
16. Joerg Schulenburg. Compute-Server Kautz. `http://www-e.uni-magdeburg.de/urzs/kautz`.
17. SiCortex. High Capability System : SC5832. `http://sicortex.com/products/high_capability_system_sc5832`.